Campus de l'Université de Montréal
C.P. 6128, succursale Centre-ville
Montréal QC H3C 3J7 Canada

Telephone (514) 343-7575
Telecopieur (514) 343-7121

crt@crt.umontreal.ca
www.crt.umontreal.ca

CRT-2002-37

Centre de recherche sur les transports (C.R.T.)
Centre for Research on Transportation

A TRUST-REGION METHOD FOR NONLINEAR BILEVEL PROGRAMMING: ALGORITHM AND COMPUTATIONAL EXPERIENCE

by

Benoît Colson
Patrice Marcotte
Gilles Savard

October 2002

PUBLICATION
CRT-2002-37

A TRUST-REGION METHOD FOR NONLINEAR BILEVEL PROGRAMMING: ALGORITHM AND COMPUTATIONAL EXPERIENCE*

by

Benoît Colson[1]
Patrice Marcotte[2]
Gilles Savard[3]

* This research was supported in part by the Ministère de la Communauté Française (Belgium), the Natural Sciences and Engineering Research Council of Canada (NSERC), and by the Fonds pour la Formation de Chercheurs et l'Aide à la Recherche (FCAR).

1 Département de Mathématiques, Facultés Universitaires Notre-Dame de la Paix, Rempart de la Vierge, 8, B-5000 Namur, Belgique <bc@math.fundp.ac.be>

2 Département d'informatique et de recherche opérationnelle and Centre de recherche sur les transports, Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, Canada H3C 3J7 <patricem@crt.umontreal.ca>

3 Département de mathématiques et de génie industriel, École Polytechnique de Montréal, C.P. 6079, succursale Centre-ville, Montréal, Canada <gilles@crt.umontreal.ca>

Abstract

We consider the approximation of nonlinear bilevel mathematical programs by solvable programs of the same type, i.e., bilevel programs involving linear approximations of the upper-level objective and all constraint-defining functions, as well as a quadratic approximation of the lower-level objective. We describe the main features of the algorithm and the resulting software. Preliminary numerical experiments tend to confirm the remarkable behavior of the method.

Keywords : bilevel programming, nonlinear programming, trust-region methods, approximation, numerical results.

Résumé

Nous considérons l'approximation de programmes mathématiques à deux niveaux par des problèmes résolubles du même type, c'est-à-dire des problèmes bi-niveaux avec une approximation linéaire de l'objectif de premier niveau et de toutes les contraintes et une approximation quadratique de la fonction objective de second niveau. Nous décrivons les principales caractéristiques de l'algorithme et le logiciel qui l'accompagne. Les résultats numériques préliminaires semblent confirmer le caractère remarquable de la méthode.

Mots clés : programmation bi-niveaux, programmation non linéaire, méthodes de région de confiance, approximation, résultats numériques.

103-06395/

Dépôt légal – Bibliothèque nationale du Québec
Bibliothèque nationale du Canada, 2002

1 Introduction

Bilevel programs, which are closely related to leader-follower (Stackelberg) games in economics, describe hierarchical structures that occur naturally in real-world situations: management (network facility location, environmental regulation, credit allocation, energy policy), economic planning (social and agricultural policies, electric power pricing, oil production), engineering design (optimal structure and shape) and optimal control, to cite but a few (see Section 2 in [47], Section 4 in [68] and Part III in [7]). Another important area of application is transportation planning and modeling (see [53] for a review). Mathematically, the general problem we address can be stated as follows:

$$\begin{aligned}
\min_{x_1,x_2} \quad & F(x_1,x_2) && \text{(1a)}\\
\text{s.t.} \quad & G(x_1,x_2) \quad \leq 0 && \text{(1b)}\\
& x_2 \in \arg\min_{x_2'} \quad f(x_1,x_2') && \text{(1c)}\\
& \text{s.t.} \quad g(x_1,x_2') \leq 0, && \text{(1d)}
\end{aligned}$$

where $F : \mathbb{R}^{n_1 \times n_2} \to \mathbb{R}$, $f : \mathbb{R}^{n_1 \times n_2} \to \mathbb{R}$, $G : \mathbb{R}^{n_1 \times n_2} \to \mathbb{R}^{m_1}$ and $g : \mathbb{R}^{n_1 \times n_2} \to \mathbb{R}^{m_2}$ are twice continuously differentiable functions. Our algorithm is designed for the case where G depends solely on the upper-level variable x_1.

The main feature of problem (1) is that it involves two mathematical programs, the solution of one being part of the constraints of the other. Due to the implicit dependence of the lower-level solution on the upper-level decision vector, bilevel programs are intrinsically hard, and the early contributions in the literature are limited to particular subclasses possessing properties such as linearity or bilinearity, which allow the development of efficient approaches.

Our approach consists in approximating a nonlinear bilevel program by a solvable problem of the same type, namely, a bilevel program of the linear-quadratic type. Although dissimilar on the surface, our technique is somewhat related to the trust-region approach developed by Scholtes and Stöhr [60] for solving mathematical programs with equilibrium constraints. However, unlike their algorithm, ours preserves the original non-convex bilevel structure of the original program. One key feature of the method is that it combines two ways of thought in mathematical optimization, namely combinatorial optimization (mixed-integer programming) and nonlinear programming (trust regions, descent). The convergence process is monitored by a trust-region mechanism which ensures that any progress obtained while minimizing the linear-quadratic model is compared to that achieved for the original problem. This allows, as new approximations are computed, efficient control and refinement of the region under consideration.

This paper is organized as follows. First, we provide a short survey of existing work related to bilevel programming and more specifically to its nonlinear instances. This is the subject of Section 2 and is based on previous work by the first author [25]. In Section 3 we recall the basic trust-region framework before we describe our algorithm for solving nonlinear bilevel programming problems. Implementation details are provided in Section 4 together with an analysis of the numerical results. We conclude by discussing extensions of our methodology to mathematical programs with equilibrium constraints.

2 Bilevel programming: a short bibliographical review

The usefulness of bilevel programming in modeling hierarchical decision processes and engineering design problems prompted researchers, in the nineteen eighties, to pay close attention to bilevel programs. Their formulation can be traced back to the 1973 article by Bracken and McGill [15], while Candler and Norton [22] were the first to use the terms bilevel and multilevel programming. A bibliographical review on the subject was written by Kolstad [47] as early as 1985, while the

most complete survey is that of Vicente and Calamai [68], whose electronic version is periodically updated. Other reference works can be found in the papers of Anandalingam and Friesz [4] and the books of Shimizu, Ishizuka and Bard [62] and Bard [7].

Bilevel programming problems are difficult to solve. Even the simplest instance – in which all functions are linear – is strongly NP-hard, as was shown by Hansen, Jaumard and Savard [39]. More recently, Vicente, Savard and Júdice [70] showed that checking local optimality for a linear bilevel program is also NP-hard. It is therefore no surprise that most algorithmic research to date has focused on the simplest cases of bilevel programs, that is problems having nice properties such as linear, quadratic or convex objective and/or constraint functions (see Bard [6] for the study of the convex case). In particular, *linear bilevel problems* have received much attention for the last two decades. Linear bilevel programs have the property (see Savard [57]) that the solution set, whenever it is nonempty, contains at least one vertex of the polyhedron

$$\Omega = \{(x_1, x_2) : \ G(x_1, x_2) \leq 0 \text{ and } g(x_1, x_2) \leq 0\}.$$

A wide class of methods is based on vertex enumeration (see *e.g.* Candler and Townsley [23], Bialas and Karwan [13], Tuy, Migdalas and Värbrand [67]). Other approaches are based on replacing the lower-level problem by its primal-dual optimality conditions. This operation reduces the original problem to a single-level program involving disjunctive constraints. Other branch-and-bound algorithms have been developed by Bard and Falk [8] and Fortuny-Amat and McCarl [36]. Combining branch-and-bound, monotonicity principles and penalization, Hansen, Jaumard and Savard [39] have developed a code capable of solving medium-size instances of linear bilevel programs. Alternatively, Bialas, Karwan and Shaw [14] and Júdice and Faustino [45, 46] based their solution approach on complementary pivoting theory. Comprehensive reviews related to that particular class of problems may be found in Hsu and Wen [40, 71] and Ben-Ayed [10].

Several algorithms have been proposed for *nonlinear bilevel programming*, under various assumptions. Roughly speaking, methods for solving the general nonlinear case may belong to one of three categories: enumeration (branch-and-bound), descent and penalty methods.

The abovementioned *branch-and-bound* approach can be extended to cover the situation where the lower-level program is convex and quadratic in the variable x_2. In this case, one replaces the lower-level problem by the equivalent Karush-Kuhn-Tucker system, whose equations and inequations are linear, with the exception of the complementarity constraint. We then obtain:

$$\begin{aligned}
\min_{x_1, x_2, \lambda} \quad & F(x_1, x_2) && && \text{(2a)}\\
\text{s.t.} \quad & G(x_1, x_2) && \leq 0, && \text{(2b)}\\
& g(x_1, x_2) && \leq 0, && \text{(2c)}\\
& \lambda_i && \geq 0, \ i = 1, \ldots, m_2 && \text{(2d)}\\
& \lambda_i g_i(x_1, x_2) && = 0, \ i = 1, \ldots, m_2 && \text{(2e)}\\
& \nabla_{x_2} \mathcal{L}(x_1, x_2, \lambda) && = 0, && \text{(2f)}
\end{aligned}$$

where[1] $\mathcal{L}(x_1, x_2, \lambda) = f(x_1, x_2) + \sum_{i=1}^{m_2} \lambda_i g_i(x, y)$. The complementary slackness conditions (2e) being a difficult type of constraint to deal with, branch-and-bound methods may be used to defer the introduction of such constraints in the solution process for as long as possible. This is the basic idea underlying the approaches of Edmunds and Bard [29] and Al-Khayyal, Horst and Pardalos [3].

Some authors have proposed *descent methods* for solving bilevel programs. Assuming that for any x_1 the optimal solution x_2^* of the lower-level problem is unique and defines x_2 as an implicit function of x_1, problem (1) may be viewed solely in terms of the upper-level variables $x_1 \in \mathbb{R}^{n_1}$.

[1] If the convexity assumption is discarded, the above formulation is that of an MPEC (Mathematical Program with Equilibrium Constraints).

Given a feasible point x_1, an attempt is made to find a direction $d \in \mathbb{R}^{n_1}$ along which the upper-level objective decreases. A new point $x_1 + \alpha d$ ($\alpha > 0$) is computed so as to ensure a reasonable decrease in F while maintaining feasibility for the bilevel problem. However, a major issue is the availability of the gradient (or the sub-gradient) of the upper-level objective, $\nabla_{x_1} F(x_1, x_2^*)$, at a feasible point. Applying the chain rule we have, whenever $\nabla_{x_1} x_2^*$ is well defined:

$$\nabla_{x_1} F(x_1, x_2^*) = \nabla_{x_1} F(x_1, x_2) + \nabla_{x_2} F(x_1, x_2)\, \nabla_{x_1} x_2^*.$$

Kolstad and Lasdon [48] have proposed a method for approximating this term. A slightly different approach is that of Savard and Gauvin [58], for problems where no upper-level constraints of the type (1b) are present and where the lower-level constraints are rewritten in the following way:

$$\begin{aligned} g_i(x_1, x_2) &\leq 0, \quad i \in I, \\ g_j(x_1, x_2) &= 0, \quad j \in J. \end{aligned}$$

They first show that an upper-level descent direction at a given point x_1 is a vector $d \in \mathbb{R}^{n_1}$ such that

$$\nabla_{x_1} F(x_1, x_2^*)\, d + \nabla_{x_2} F(x_1, x_2^*)\, w(x_1, d) < 0, \tag{3}$$

where $w \in \mathbb{R}^{n_2}$ is a solution of the program

$$\begin{array}{ll} \min_w & (d^T, w^T)\, \nabla^2_{x_1 x_2} \mathcal{L}(x_1, x_2^*, \lambda)\, (d, w) \\ \text{s.t.} & \nabla_{x_2} g_i(x_1, x_2^*) w \leq -\nabla_{x_1} g_i(x_1, x_2^*) d, \; i \in I(x_1), \\ & \nabla_{x_2} g_j(x_1, x_2^*) w = -\nabla_{x_1} g_j(x_1, x_2^*) d, \; j \in J, \\ & \nabla_{x_2} f(x_1, x_2^*) w = -\nabla_{x_1} f(x_1, x_2^*) d + \nabla_{x_1} \mathcal{L}(x_1, x_2^*, \lambda) d, \end{array} \tag{4}$$

where $I(x_1) = \{i \in I : g_i(x_1, x_2^*) = 0\}$ and

$$\mathcal{L}(x_1, x_2, \lambda) = f(x_1, x_2) + \sum_{i \in I(x_1) \cup J} \lambda_i g_i(x_1, x_2)$$

is the Lagrangian of the lower-level problem. The steepest descent direction among those defined by (3) may be shown to coincide with the optimal solution of the *linear-quadratic bilevel program*

$$\begin{array}{ll} \min_d & \nabla_{x_1} F(x_1, x_2^*) d + \nabla_{x_2} F(x_1, x_2^*) w(x_1, d) \\ \text{s.t.} & \|d\| \leq 1, \\ & w(x_1, d) \text{ solves the quadratic program (4),} \end{array}$$

for which exact algorithms exist (see *e.g.* Bard and Moore [9], Jaumard, Savard and Xiong [43]). We also mention the work of Falk and Liu [31] who present a bundle method where the decrease of the upper-level objective is adapted according to subdifferential information obtained from the lower-level problem. They call the resulting setup a *leader predominate algorithm* in view of the role of the leader in the sequential decision making process.

Finally, *penalty methods* constitute a third category of algorithms. Because of the inherent difficulty of the nonlinear case, these methods are generally limited to computing stationary points and local minima. An initial step in this direction was achieved by Aiyoshi and Shimizu [1, 2, 61]. Their approach consists in replacing the lower-level problem (1c)-(1d) by the penalized problem

$$\min_{x_2} \quad p(x_1, x_2, r) = f(x_1, x_2) + r\, \phi(g(x_1, x_2)), \tag{5}$$

where r is a positive scalar, ϕ is a continuous interior penalty function that satisfies

$$\begin{array}{ll} \phi(g(x_1, x_2)) > 0 & \text{if } x_2 \in \text{int } S(x_1), \\ \phi(g(x_1, x_2)) \to +\infty & \text{if } x_2 \to \text{bd } S(x_1), \end{array} \tag{6}$$

and int $S(x_1)$ and bd $S(x_1)$ denote the interior and the boundary of $S(x_1) = \{x_2 : g(x_1, x_2) \leq 0\}$, respectively. Problem (1) is then transformed into:

$$\begin{array}{lll} \min_{x_1} & F(x_1, x_2^*(x_1, r)) & \\ \text{s.t.} & G(x_1, x_2^*(x_1, r)) & \leq 0, \\ & p(x_1, x_2^*(x_1, r), r) & = \min_{x_2} p(x_1, x_2, r). \end{array} \tag{7}$$

Shimizu and Aiyoshi [61] proved that the sequence $\{(x_1^k, x_2^*(x_1^k, r^k))\}$ of optimal solutions to (7) converges to the solution of (1). The main drawback of this method is that solving (7) for a fixed value of r requires the solution of the unconstrained lower-level problem at every update of the upper-level variables. As might be expected, this significantly slows down the convergence process. Ishizuka and Aiyoshi [42] proposed a double penalty method in which both objective functions (1a) and (1c) are penalized. They still use the augmented lower-level objective (5) and the penalty function ϕ characterized by (6) but replace the lower-level problem by its stationarity condition $\nabla_{x_2} p(x_1, x_2, r) = 0$, thus transforming (1) into the single-level program

$$\begin{array}{lll} \min_{x_1, x_2} & F(x_1, x_2) & \\ \text{s.t.} & G(x_1, x_2) & \leq 0, \\ & \nabla_{x_2} p(x_1, x_2, r) & = 0, \\ & g(x_1, x_2) & < 0. \end{array} \tag{8}$$

Note that the last constraint restrains the domain of the function p. For a given r, problem (8) is solved using a second penalty function applied to the upper-level objective $F(x_1, x_2)$. A more recent contribution is proposed in Case's PhD thesis [24] and follows up on ideas of Bi, Calamai and Conn [12], who themselves extend a technique they proposed earlier for linear bilevel programs [11]. Their approach focuses on a problem of type (2), that is, a bilevel program for which the lower-level problem has been replaced by its Karush-Kuhn-Tucker conditions. Their method involves a penalty function of the form

$$p(x_1, x_2, \lambda, \mu) = F(x_1, x_2) + \mu\, \nu(x_1, x_2, \lambda), \tag{9}$$

where μ is a positive *penalty parameter* and the upper-level objective $F(x_1, x_2)$ is augmented with a weighted, nonnegative penalty function associated with the current iterate. More precisely, Case *et al* [24] build a penalty function $\nu(x_1, x_2, \lambda)$ with respect to the l_1 norm, defined as the sum of the terms associated with each constraint of the single-level problem (2). The resulting algorithm involves the minimization of the penalty function $p(x_1, x_2, \lambda, \mu)$ for a fixed value of μ. In view of the complex structure of the latter function, the authors use a trust-region method, *i.e.*, they minimize a simplified version of p whose local behavior is expected to model that of p. The trust-region model adopted by [24] corresponds to replacing each component function of $p(x_1, x_2, \lambda, \mu)$ by its second-order Taylor expansion around the current iterate.

We conclude this section by mentioning the relationship between bilevel programming problems and another important class of problems, namely *Mathematical Programs with Equilibrium Constraints*, or MPECs. The word *equilibrium* frequently refers to the formulation of the lower-level problem as a finite-dimensional variational inequality. This allows the mathematical representation of phenomena which may be encountered in engineering, physics, chemistry or economics. For a given function $F : \mathbb{R}^{n_1+n_2} \to \mathbb{R}$, the general form of an MPEC is as follows

$$\begin{array}{ll} \min_{x_1, x_2} & F(x_1, x_2) \\ \text{s. t.} & (x_1, x_2) \in Z \text{ and } x_2 \in \mathcal{S}(x_1), \end{array} \tag{10}$$

where $Z \subseteq \mathbb{R}^{n_1+n_2}$ is a nonempty closed set and $\mathcal{S}(x_1)$ is the solution set of a variational inequality (or VI) parameterized in x_1, that is,

$$\begin{array}{lll} x_2 \in \mathcal{S}(x_1) \Leftrightarrow & x_2 \in C(x_1) \text{ and} \\ & (v - x_2)^T \phi(x_1, x_2) \geq 0 \; \forall v \in C(x_1), \end{array} \tag{11}$$

where $C(x_1)$ is a closed convex subset of $\mathbb{R}^{n_2}$ and $\phi : \mathbb{R}^{n_1+n_2} \to \mathbb{R}^{n_2}$. As for bilevel problems, the terms upper-level and lower-level variables are also used to designate x_1 and x_2 respectively. If the mapping $\phi(x_1, \cdot)$ is the partial gradient map of a real-valued C^1 function $f : \mathbb{R}^{n_1+n_2} \to \mathbb{R}$, that is,

$$\phi(x_1, x_2) = \nabla_{x_2} f(x_1, x_2),$$

then, for any fixed x_1, the VI (11) defines the set of stationarity conditions of the optimization problem

$$\begin{array}{ll} \min_{x_2} & f(x_1, x_2) \\ \text{s.t.} & x_2 \in C(x_1). \end{array} \tag{12}$$

Moreover, if $C(x_1)$ takes the form

$$C(x_1) = \{x_2 \ : \ g(x_1, x_2) \leq 0\},$$

then problem (12) is nothing but the lower-level problem (1c)-(1d). This shows that MPECs subsume bilevel programs with convex lower-level problems. Conversely, an MPEC can be formulated as a bilevel program by reformulated the lower-level VI as an optimization problem. The optimization problem can, for instance, be constructed around the gap function (see *e.g.* Auslender [5] or Fukushima [37]), or by simply minimizing the complementarity slackness term.

While many active set approaches have been proposed for solving MPECs (see Bard [7]), more recent methods deal with constraint regularization (*e.g.* Facchinei, Jian and Qi [30], Fukushima and Pang [38], or Scholtes [59]), implicit programming techniques (Outrata, Kocvara and Zowe [55]), or nonlinear programming techniques (filter methods: Fletcher and Leyffer [32, 33], sequential quadratic programming: see Jiang and Ralph [44], Ralph [56] and Fletcher *et al* [34]). We refer to the monographs of Luo, Pang and Ralph [51] and Outrata, Kocvara and Zowe [55] for a comprehensive analysis of MPECs.

3 An approximation algorithm for bilevel programming problems

In this section, we give an overview of the trust-region framework and its adaptation to bilevel programming, in the absence of joint upper-level constraints, *i.e.*, constraints unrelated to the lower level and involving both upper and lower-level decision variables. In the third subsection, we discuss a modification that allows to deal with such constraints.

3.1 The trust-region framework

We briefly recall the main features of *trust-region methods* for minimizing an unconstrained function ϕ. At iteration k of the algorithm, we approximate the optimization problem by a *model* built around the best point found so far, denoted by $x^{(k)}$, and request that the model m_k coincide with the objective ϕ at $x^{(k)}$, *i.e.*,

$$m_k(x^{(k)}) = \phi(x^{(k)}).$$

More generally the model is assumed to be a reasonable approximation of ϕ within $\mathcal{B}(x^{(k)}, \Delta_k)$, the *trust-region* of radius Δ_k centered around $x^{(k)}$. A new – hopefully better – point is then determined through the search for an approximation $s^{(k)}$ to the optimal solution of the *trust-region subproblem*

$$\min_{\|s\| \leq \Delta_k} \ m_k(x^{(k)} + s). \tag{13}$$

Next, one checks whether $m_k(x^{(k)} + s^{(k)})$ is sufficiently close to $\phi(x^{(k)} + s^{(k)})$, in which case the model decrease is deemed a good estimator of the actual decrease of the objective value. To achieve this, one compares the *actual reduction*

$$\phi(x^{(k)}) - \phi(x^{(k)} + s^{(k)}) \tag{14}$$

of the objective function value with the *predicted reduction*

$$m_k(x^{(k)}) - m_k(x^{(k)} + s^{(k)}) \tag{15}$$

given by the model m_k. If the actual reduction is smaller than a fixed ratio of the predicted reduction, then the model is deemed inadequate within $\mathcal{B}(x^{(k)}, \Delta_k)$ and the trust-region radius is reduced, in order to improve the local accuracy of the model. If the actual reduction exceeds a certain fixed ratio of the predicted reduction, then $x^{(k)}$ is updated to $x^{(k+1)} := x^{(k)} + s^{(k)}$, a new model m_{k+1} is built and the trust-region radius $\Delta_{k+1} \geq \Delta_k$ is updated. This can be summarized in the following setup, where η_1, η_2, γ_1 and γ_2 are fixed parameters such that

$$0 < \eta_1 \leq \eta_2 < 1 \text{ and } 0 < \gamma_1 < 1 < \gamma_2. \tag{16}$$

At iteration k, given $x^{(k)} \in \mathbb{R}^n$, $\Delta_k > 0$ and $m_k : \mathbb{R}^n \to \mathbb{R}$:

1. Determine a vector $s^{(k)}$ with
   (a) $\|s^{(k)}\| \leq \Delta_k$, and
   (b) $m_k(x^{(k)} + s^{(k)}) < m_k(x^{(k)})$.
   If no such vector exists, then stop.
2. Compute ρ_k, the ratio of achieved versus predicted reduction:
$$\rho_k := \frac{\phi(x^{(k)}) - \phi(x^{(k)} + s^{(k)})}{m_k(x^{(k)}) - m_k(x^{(k)} + s^{(k)})}.$$
3. If $\rho_k < \eta_1$, then set $x^{(k+1)} := x^{(k)}$, $\Delta_{k+1} := \gamma_1 \Delta_k$,
   else if $\eta_1 \leq \rho_k < \eta_2$ then set $x^{(k+1)} := x^{(k)} + s^{(k)}$, $\Delta_{k+1} := \Delta_k$,
   else set $x^{(k+1)} := x^{(k)} + s^{(k)}$, $\Delta_{k+1} := \gamma_2 \Delta_k$.
4. Determine m_{k+1}.

An iteration k is said to be *successful* if $\rho_k \geq \eta_1$.

A sequence $\{x^{(k)}\}$ is called a *trust-region sequence* if it can be generated by the above method, starting at an initial vector $x^{(0)}$. It clearly defines a descent algorithm, since $\phi(x^{(k+1)}) \leq \phi(x^{(k)})$ for every $k \in \mathbb{N}$. Therefore, this strategy can be used for the minimization of an arbitrary function ϕ. For an in-depth analysis of trust-region methods we refer to the monograph by Conn, Gould and Toint [27].

3.2 Description of the algorithm

The framework of the previous subsection can be adapted to find (local) optima of bilevel programs without joint upper-level constraints, *i.e.*, when the function G depends solely on the upper-level vector x_1. Once an initial solution is available (see details below), we build a *linear-quadratic* model of problem (1) around local linear approximations of F, G and g, and a quadratic approximation of the lower-level objective f. If $(\bar{x}_1, \bar{x}_2)$ denotes the current iterate, we set up the model functions:

$$F_m(x_1, x_2) = F(\bar{x}_1, \bar{x}_2) + c_1^T(x_1 - \bar{x}_1) + c_2^T(x_2 - \bar{x}_2), \tag{17}$$

$$G_m(x_1) = G(\bar{x}_1) + A_1(x_1 - \bar{x}_1), \tag{18}$$

$$\begin{aligned} f_m(x_1, x_2) = f(\bar{x}_1, \bar{x}_2) &+ d_1^T(x_1 - \bar{x}_1) + d_2^T(x_2 - \bar{x}_2) \\ &+ \tfrac{1}{2}(x_1 - \bar{x}_1)^T Q_{11}(x_1 - \bar{x}_1) \\ &+ (x_1 - \bar{x}_1)^T Q_{12}(x_2 - \bar{x}_2) \\ &+ \tfrac{1}{2}(x_2 - \bar{x}_2)^T Q_{22}(x_2 - \bar{x}_2), \end{aligned} \tag{19}$$

$$g_m(x_1, x_2) = g(\bar{x}_1, \bar{x}_2) + H_1(x_1 - \bar{x}_1) + H_2(x_2 - \bar{x}_2), \tag{20}$$

where

$$c_1 = \nabla_{x_1} F(\bar{x}_1, \bar{x}_2) \in \mathbb{R}^{n_1},\ c_2 = \nabla_{x_2} F(\bar{x}_1, \bar{x}_2) \in \mathbb{R}^{n_2},$$

$$A_1 = J_{x_1} G(\bar{x}_1) \in \mathbb{R}^{m_1 \times n_1},$$

$$d_1 = \nabla_{x_1} f(\bar{x}_1, \bar{x}_2) \in \mathbb{R}^{n_1},\ d_2 = \nabla_{x_2} f(\bar{x}_1, \bar{x}_2) \in \mathbb{R}^{n_2},$$

$$H_1 = J_{x_1} g(\bar{x}_1, \bar{x}_2) \in \mathbb{R}^{m_2 \times n_1},\ H_2 = J_{x_2} g(\bar{x}_1, \bar{x}_2) \in \mathbb{R}^{m_2 \times n_2},$$

$$Q_{11} = \nabla^2_{x_1 x_1} f(\bar{x}_1, \bar{x}_2) \in \mathbb{R}^{n_1 \times n_1},\ Q_{12} = \nabla^2_{x_1 x_2} f(\bar{x}_1, \bar{x}_2) \in \mathbb{R}^{n_1 \times n_2},$$

$$Q_{21} = \nabla^2_{x_2 x_1} f(\bar{x}_1, \bar{x}_2) \in \mathbb{R}^{n_2 \times n_1},\ Q_{22} = \nabla^2_{x_2 x_2} f(\bar{x}_1, \bar{x}_2) \in \mathbb{R}^{n_2 \times n_2}.$$

The above notation yields the *linear-quadratic* bilevel program:

$$\min_{x_1, x_2} \quad F_m(x_1, x_2) \tag{21a}$$

$$\text{s.t.} \quad G_m(x_1) \quad \leq 0 \tag{21b}$$

$$\min_{x_2} \quad f_m(x_1, x_2) \tag{21c}$$

$$\text{s.t.} \quad g_m(x_1, x_2) \leq 0. \tag{21d}$$

Due to its combinatorial structure, problem (21) can be reformulated as a mixed-integer program (MIP) and solved using either specialized algorithms (see Jaumard, Savard and Xiong [43]) or an off-the-shelf MIP software. To transform (21) into an equivalent MIP, we first remove the constant terms from (17) and (19) and set

$$\bar{G} = G(\bar{x}_1) - A_1 \bar{x}_1,$$

and

$$\bar{g} = g(\bar{x}_1, \bar{x}_2) - H_1 \bar{x}_1 - H_2 \bar{x}_2.$$

The upper and lower-level functions are then expressed as

$$F_{m'}(x_1, x_2) = c_1^T x_1 + c_2^T x_2,$$

$$G_{m'}(x_1) = A_1 x_1,$$

$$f_{m'}(x_1, x_2) = r_1^T x_1 + r_2^T x_2 + \frac{1}{2}(x_1^T\ x_2^T) \begin{pmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix},$$

$$g_{m'}(x_1, x_2) = H_1 x_1 + H_2 x_2,$$

where

$$r_1 = d_1 - Q_{21}^T \bar{x}_2 - Q_{11}^T \bar{x}_1 \in \mathbb{R}^{n_1}$$

and

$$r_2 = d_2 - Q_{12}^T \bar{x}_1 - Q_{22}^T \bar{x}_2 \in \mathbb{R}^{n_2}.$$

The problem (21) can then be rewritten as:

$$
\begin{array}{lll}
\min_{x_1} & F_{m'}(x_1, x_2) & \\
\text{s.t.} & G_{m'}(x_1) & \leq -\bar{G} \\
 & \min_{x_2} & f_{m'}(x_1, x_2) \\
 & \text{s.t.} & g_{m'}(x_1, x_2) \leq -\bar{g}.
\end{array}
$$

Incorporating the trust-region constraint with respect to the upper-level vector x_1, we derive an equivalent MIP:

$$
\begin{aligned}
\min \quad & c_1^T x_1 + c_2^T x_2 && \text{(22a)} \\
\text{s.t.} \quad & \|x_1 - x_1^{(k)}\| \leq \Delta_k, && \text{(22b)} \\
& A_1 x_1 \leq -\bar{G}, && \text{(22c)} \\
& H_1 x_1 + H_2 x_2 \leq -\bar{g}, && \text{(22d)} \\
& \lambda \geq 0 \in \mathbb{R}^{m_2}, && \text{(22e)} \\
& \lambda_i \leq M z_i, \; i = 1, \ldots, m_2, && \text{(22f)} \\
& [-\bar{g} - H_1 x_1 - H_2 x_2]_i \leq M(1 - z_i), \; i = 1, \ldots, m_2, && \text{(22g)} \\
& z_i \in \{0, 1\}, \; i = 1, \ldots, m_2, && \text{(22h)} \\
& \frac{1}{2}(Q_{12}^T + Q_{21}) x_1 + Q_{22} x_2 + H_2^T \lambda + r_2 = 0. && \text{(22i)}
\end{aligned}
$$

The trust-region constraint is expressed as (22b), while (22c) represents the primal feasibility of the upper-level problem. Primal feasibility of the lower-level problem is ensured by (22d); dual feasibility corresponds to (22e) and (22i). Finally, the complementarity constraint

$$
\lambda_i [-\bar{g} - H_1 x_1 - H_2 x_2]_i = 0, \; i = 1, \ldots, m_2
$$

is linearized and replaced by (22f), (22g) and (22h), where the parameter M is a 'large number' that ensures that the MIP is equivalent to the model bilevel program.

Let (x_1^m, x_2^m) denote a global solution of (22). In order to evaluate the true value of this solution, one must compute the lower-level reaction to x_1^m, *i.e.*, the optimal solution of

$$
\begin{aligned}
\min_{x_2} \quad & f(x_1^m, x_2) && \text{(23a)} \\
\text{s.t.} \quad & g(x_1^m, x_2) \leq 0, && \text{(23b)}
\end{aligned}
$$

which we denote by x_2^*. Since a *global* solution of (23) is required, the lower-level problem will be assumed to be convex, unless one is willing to solve to optimality a non-convex program, using global optimization techniques. This completes the description of the trust-region subproblem which, starting from a given point $(\bar{x}_1, \bar{x}_2)$, generates a solution (x_1^m, x_2^*) of a model of (1) around $(\bar{x}_1, \bar{x}_2)$. We are now in position to summarize the complete algorithm.

1. **Initialization.** Initial values are provided for the trust-region radius Δ_0, the parameters η_1, η_2, γ_1 and γ_2 as in (16), M, $\Delta_{\min}$, $k_{\max}$ and ε. It is assumed that an initial feasible point $(x_1^{(0)}, x_2^{(0)})$ is available. This point may for instance correspond to the *high point*, that is, the solution of the single-level program:

$$
\begin{array}{lll}
\min_{x_1, x_2} & F(x_1, x_2) & \\
\text{s.t.} & G(x_1) \leq 0 & \\
 & g(x_1, x_2) \leq 0. &
\end{array}
\qquad \text{(24)}
$$

The iteration counter k is set to 0.

2. **Computation of the model.** Construct $\bar{G}$, $\bar{g}$ and the non-constant terms in the model functions F_m, G_m, f_m and g_m.

3. **Solving the subproblem.** Solve problems (22) and (23) for their respective solutions (x_1^m, x_2^m) and x_2^*.

4. **Updating radius and iterate.** Compute the ratio of achieved versus predicted reduction

$$\rho_k = \frac{F(x_1^{(k)}, x_2^{(k)}) - F(x_1^m, x_2^*)}{F_m(x_1^{(k)}, x_2^{(k)}) - F_m(x_1^m, x_2^m)}.$$

   If $\rho_k < \eta_1$, the model is deemed inaccurate: set $(x_1^{(k+1)}, x_2^{(k+1)}) := (x_1^{(k)}, x_2^{(k)})$ and $\Delta_{k+1} := \gamma_1 \Delta_k$.

   Otherwise, set $(x_1^{(k+1)}, x_2^{(k+1)}) := (x_1^m, x_2^*)$ and $\Delta_{k+1} := \gamma_2 \Delta_k$ if $\rho_k \geq \eta_2$.

5. **Stopping criteria.** The optimization process is stopped if one of the following criteria is met:

   - $\|x_1^{(k+1)} - x_1^{(k)}\| < \varepsilon$ after a successful iteration,
   - the actual reduction equals the predicted one and the latter is small,
   - a predetermined number of consecutive unsuccessful iterations has been reached,
   - the trust-region radius becomes smaller than some predetermined threshold ($\Delta_{k+1} < \Delta_{\min}$),
   - the number of iterations exceeds some predetermined number (*i.e.* $k + 1 > k_{\max}$).

   If none of the above conditions is satisfied, set $k := k + 1$ and return to step 2.

3.3 Joint upper-level constraints

Joint upper-level constraints $G(x_1, x_2) \leq 0$ are transparent to the lower-level 'player' and must naturally be induced by the upper-level vector x_1. Finding a feasible point that respects these constraints and is compatible with the lower-level reaction is, theoretically, as difficult as solving a bilevel program. Actually, these constraints can make the feasible set of the bilevel program disconnected. Since these constraints are not in the direct control of the leader player, their inclusion in the trust-region model could lead to infeasible solutions, once the lower-level reaction is properly taken into account. We deal with this situation by simply penalizing away joint upper-level constraints into the objective, using a quadratic penalty function.

4 Computational experience

BIPA (**B**ilevel **P**rogramming with **A**pproximation methods) is a C ANSI code for solving problems of type (1), based on the above-described methodology. The software is designed around existing and efficient tools for addressing subproblems (22) and (23). This enables us to focus our programming effort on both the approximation and trust-region aspects. The MIP (22) is solved using the `CPXmipopt` routine from the CPLEX Callable Library [41], while problem (23) is solved using the C ANSI version of Spellucci's DONLP2 code (see [65], [66] for the underlying algorithm and [64] for DONLP2 users guide). An in-depth description of BIPA is provided in [26].

We tested BIPA on a set of representative instances, which are listed in Table 1 below. Some problems involving linear or quadratic functions are borrowed from the literature. Others are new problems created by customizing some of the former ones through the introduction of nonlinearities. A third set of test problems is based on two applications of bilevel programming to transportation modeling and planning, namely a continuous variant of the network design problem and a toll-setting problem. We also tested BIPA on a problem introduced by Outrata [54] in a paper dedicated to MPECs. A complete description of all our test problems may be found in the appendix. Note that some of the problems do not satisfy the assumption on the convexity of the lower-level problem.

Problem type	Problem name	n_1	n_2	m_1	m_2	Reference
Quadratic-quadratic	`Bard88Ex1`	1	1	1	4	Bard [6]
	`Bard88Ex2`	4	4	9	12	Bard [6]
	`Bard88Ex3`	2	2	3	4	Bard [6]
	`Dempe92`	1	1	0	1	Dempe [28]
	`DeSilva78`	2	2	0	4	DeSilva [63]
	`FalkLiu95`	2	2	0	4	Falk and Liu [31]
	`ShimAiyo81P1`	1	1	3	3	Shimizu and Aiyoshi [61]
Nonlinear	`BIPA1`	1	1	3	3	new problem
	`BIPA2`	1	1	1	4	new problem
	`BIPA3`	1	1	2	2	new problem
	`BIPA4`	1	1	2	2	new problem
	`BIPA5`	1	2	1	6	new problem
Network design	`NDP1`	5	5	5	8	see appendix
	`NDP2`	5	5	5	8	see appendix
Toll-setting	`TOLL1`	3	8	3	13	Labbé *et al* [49]
	`TOLL2`	3	18	3	28	new problem
	`TOLL3`	3	18	3	28	new problem
	`TOLL4`	2	4	2	6	Brotcorne [17]
	`TOLL5`	1	4	1	6	Brotcorne [17]
MPEC	`Outrata94P1`	1	2	2	4	Outrata [54]

Table 1: Test problems: data and references.

The values of the various parameters are reported in Table 2, where `mxnuns` denotes the maximum number of unsuccessful iterations. Trust-region parameters agree with those recommended by Conn, Gould and Toint [27], with the exception of the initial trust-region radius, for which we could observe that a larger value often leads to an increase in the number of iterations. Optimal values for $k_{\max}$ and $\Delta_{\min}$ proved to be non-critical. Indeed, the associated stopping criteria (number of iterations too large or trust-region radius too small) were seldom met in practice. Finally, we emphasize the role of the parameter M appearing in the MIP (22). Ideally, M should be chosen as small as possible, in order that the linear programming relaxation of the MIP be as tight as possible, thus improving the quality of the lower bound and accelerating the implicit enumeration process. However, if M assumes a value that is too small, the optimal solution could be missed and, in an extreme situation, the MIP could be infeasible. Although our computational experiments showed that the value $M = 100$ was 'uniformly' suitable, this issue, which is problem dependent, would certainly deserve a more thorough analysis.

The numerical results are summarized in Table 3. All tests were performed on a Sun Ultra-10/300 computer (512M RAM). The starting points for the algorithm are given in the appendix. For the problems taken from the literature, we used the initial points provided in the relevant papers.

Initial trust-region radius	Δ_0	10.0
Other trust-region parameters	η_1	0.01
	η_2	0.90
	γ_1	0.6
	γ_2	1.4
MIP-parameter	M	1.0e+2
Stopping criteria	$k_{\max}$	50
	$\Delta_{\min}$	1.0e-6
	ε	1.0e-6
	mxnuns	5

Table 2: Parameters for numerical experiments.

Problem name	Iter.	CPU time	Objectives at solution $F(x_1^*, x_2^*)$	$f(x_1^*, x_2^*)$	Remarks
AiyoShim84P2	2	0.40	0.00	200.00	Optimal solution
Bard88Ex1	7	1.12	17.00	1.00	Optimal solution
Bard88Ex2	32	6.22	-6600.00	57.48	Optimal solution
Bard88Ex3	1	0.20	-12.68	-1.02	Optimal solution
Dempe92	6	0.94	31.25	4.00	Optimal solution
DeSilva78	13	2.10	-1.00	0.00	Opt. sol., started at $(0, 0, 1, 1)$
	7	1.23	-1.00	0.00	Opt. sol., started at $(1, 1, 1, 1)$
FalkLiu95	47	7.76	-2.25	0.00	Opt. sol., started at $(0, 0, 1, 1)$
	45	7.14	-2.25	0.00	Opt. sol., started at $(1, 1, 1, 1)$
ShimAiyo81P1	50	7.86	99.91	0.00	With penalty
BIPA1	9	1.37	229.90	0.00	With penalty
BIPA2	7	1.16	17.00	2.00	Best solution found
BIPA3	1	0.17	2.00	24.02	Best solution found
BIPA4	2	0.33	88.79	-0.77	Best solution found
BIPA5	16	2.53	2.75	0.57	Best solution found
NDP1	6	1.09	300.50	419.80	Optimal solution
NDP2	6	1.04	142.90	81.95	Optimal solution
TOLL1	10	2.36	-7.00	12.00	Optimal solution
TOLL2	9	4.19	-9.00	32.00	Optimal solution
TOLL3	10	4.48	-24.00	81.00	Optimal solution
TOLL4	8	1.32	-6.00	12.00	Optimal solution
TOLL5	7	1.16	-5.00	14.00	Optimal solution
Outrata94P1	12	1.88	3.2077	-20.5293	Optimal solution, $x_1^{(0)} = 0$
	10	1.69	3.2077	-20.5293	Optimal solution, $x_1^{(0)} = 10$

Table 3: Results with BIPA.

Otherwise, we started the algorithm from the high point (see (24)).

Generally speaking, all problems are solved in a small number of iterations. BIPA is able to find the optimal solution of all problems from the literature as well as for the NDP and TOLL problems. For problem `FalkLiu95`, our solution improves[2] on that reported in [31]. For the new problems, we report the best solution found.

For most problems the optimal solution is found very early and the subsequent iterations are used to check optimality. Even for problems requiring a larger number of iterations, good solutions were found rapidly. For instance, $F(x_1, x_2) = -6598.4687$ after only 10 iterations for problem `Bard88Ex2`, $F(x_1, x_2) = -1$ after 7 iterations for `DeSilva78` (first starting point), and $F(x_1, x_2) = -2.25$ after 20 iterations for problem `FalkLiu95`.

Toll-setting problems constitute a very particular class of problems for which specific methods are developed. Actually, building a linear-quadratic model of these problems might deteriorate their structure since it ignores the bilinear structure of the lower-level. However the algorithms performs well and produces the optimal solution with a very reasonable amount of computational work.

Since problems `ShimAiyo81P1` and `BIPA1` involve a joint upper-level constraint, namely $x_2 \leq x_1$ (see the complete formulation of both problems in the appendix), we solved them using the strategy described in Section 3.3. We added the penalty term

$$p(x_1, x_2) = L \, \frac{1}{2} \left(\max(0, x_2 - x_1)\right)^2$$

corresponding to the joint constraint to the upper-level objective, where L is a large number (we choose $L = 1000$ for the numerical tests). The optimal solution of problem `ShimAiyo81P1` is $(x_1^*, x_2^*) = (10, 10)$. After 50 iterations, the modified version of BIPA using the penalty function stops because the trust-region radius falls below 10^{-6} and finds $(x_1^*, x_2^*) = (9.991121, 10.00444)$, which is very close to the optimal solution. For `BIPA1`, our algorithm stops after 9 iterations because no further progress is achieved, the optimal solution being $(x_1^*, x_2^*) = (6.12, 4.44)$.

Some comments about problem `Outrata94P1` are in order. This problem was originally introduced by Outrata [54] and is part of a series of four problems. These problems represent MPECs where the variational inequality corresponds to the optimality conditions of the lower-level and hence they involve multipliers for the lower-level constraints. Only the first of these four problems fits our framework. The remaining three involve those multipliers in the upper-level objective, which leads to a class of problems slightly different from the one we study.

5 Comments and future work

The originality of our algorithm is twofold, in that it combines techniques from nonlinear programming (trust-region methods) and combinatorial optimization (mixed-integer programming), and preserves the bilevel structure of the approximated model. The numerical results collected so far reflect the good behavior of the algorithm and the robustness of the associated code.

At this point, we would like to make two comments about the numerical testing of bilevel programs. First, the number of test problems treated in this paper, as well as the size of these instances, is small. This is due to the fact that very few instances of large-scale, unstructured, nonlinear bilevel programs are available in the literature. Actually, some of the examples addressed in this paper are the largest having been addressed by a software that does not take into account the specific structure of the problems. While there exist generators of bilevel problems (see Calamai and Vicente [18, 19, 20], Calamai, Vicente and Júdice [21] or Vicente, Calamai and Júdice [69]), these are limited to linear or quadratic problems, and are obviously irrelevant to our approach. To our knowledge, there exist but two collections of test problems: the first is described in Chapter 9 of Floudas

[2] Note that we had to set `mxnuns` $= 6$ to be able to find the optimal solution with the second starting point.

et al [35] and contains 19 problems, most of them linear or quadratic; the second is the *MacMPEC collection* maintained by Leyffer [50], which contains about 130 MPECs, only a handful of them being equivalent to bilevel programs.

Our second comment is concerned with the lack of a proper testing environment (including a possible proper modeling language) for solving general bilevel programs. Such an environment should provide all the necessary tools for evaluating objective functions, gradients, ... and would allow to avoid coding these functions from scratch.

Finally, in view of the promising results obtained by our algorithm, we will extend the methodology to mathematical problems with (monotone) variational inequality constraints, a subclass of MPEC of great practical importance. Since no robust software is currently available for general variational inequalities, we intend to adapt DONLP2 for this purpose.

Acknowledgements

Work of the first author was supported by a grant from the *Ministère de la Communauté Française* (Belgium). The first author also thanks the *Centre de Recherche sur les Transports* for having provided a cordial environment. Finally, computational support from Serge Bisaillon and Luc Rocheleau is gratefully acknowledged.

Appendix: Test problems

Quadratic-quadratic programs

- `Bard88Ex1`: example 1 in Bard [6].

$$
\begin{array}{lll}
\min_{x_1} & F(x_1,x_2) & = (x_1-5)^2+(2x_2+1)^2 \\
\text{s.t.} & -x_1 \le 0, & \\
 & \min_{x_2} & f(x_1,x_2) = (x_2-1)^2 - 1.5x_1x_2 \\
 & \text{s.t.} & -3x_1+x_2+3 \le 0, \\
 & & x_1-0.5x_2-4 \le 0, \\
 & & x_1+x_2-7 \le 0, \\
 & & -x_2 \le 0.
\end{array}
$$

  Starting point: $(x_1,x_2) = (4,0)$.

- `Bard88Ex2`: example 2 in Bard [6].
  The original formulation of this problem involved two independent lower-level problems. We regrouped those two problems so as to form a unique lower-level problem.

$$
\begin{array}{lll}
\min_{x_1} & F(x_1,x_2) & = -(200-x_{21}-x_{23})(x_{21}+x_{23}) - (160-x_{22}-x_{24})(x_{22}+x_{24}) \\
\text{s.t.} & x_{11}+x_{12} & +x_{13}+x_{14} \le 40, \\
 & 0 \le x_{11} \le & 10, \\
 & 0 \le x_{12} \le & 5, \\
 & 0 \le x_{13} \le & 15, \\
 & 0 \le x_{14} \le & 20, \\
 & \min_{x_2} & f(x_1,x_2) = (x_{21}-4)^2+(x_{22}-13)^2+(x_{23}-35)^2+(x_{24}-2)^2 \\
 & \text{s.t.} & 0.4x_{21}+0.7x_{22}-x_{11} \le 0, \\
 & & 0.6x_{21}+0.3x_{22}-x_{12} \le 0, \\
 & & 0.4x_{23}+0.7x_{24}-x_{13} \le 0, \\
 & & 0.6x_{23}+0.3x_{24}-x_{14} \le 0, \\
 & & 0 \le x_{21} \le 20, \\
 & & 0 \le x_{22} \le 20, \\
 & & 0 \le x_{23} \le 40, \\
 & & 0 \le x_{24} \le 40.
\end{array}
$$

Starting point: $(x_{11}, x_{12}, x_{13}, x_{14}, x_{21}, x_{22}, x_{23}, x_{24}) = (5, 5, 15, 15, 0, 0, 0, 0)$.

- `Bard88Ex3`: example 3 in Bard [6].

$$\begin{array}{lll} \min_{x_1} & F(x_1, x_2) = & -x_{11}^2 - 3x_{12} - 4x_{21} + x_{22}^2 \\ \text{s.t.} & x_{11}^2 + 2x_{12} - 4 & \leq 0, \\ & -x_{11} \leq 0, & \\ & -x_{12} \leq 0, & \\ & \min_{x_2} & f(x_1, x_2) = 2x_{11}^2 + x_{21}^2 - 5x_{22} \\ & \text{s.t.} & -x_{11}^2 + 2x_{11} - x_{12}^2 + 2x_{21} - x_{22} - 3 \leq 0, \\ & & -x_{12} - 3x_{21} + 4x_{22} + 4 \leq 0, \\ & & -x_{21} \leq 0, \\ & & -x_{22} \leq 0. \end{array}$$

Starting point: $(x_{11}, x_{12}, x_{21}, x_{22}) = (0, 2, 4, 1)$.

- `Dempe92`: a problem from Dempe [28].

$$\begin{array}{lll} \min_{x_1} & F(x_1, x_2) & = (x_1 - 3.5)^2 + (x_2 + 4)^2 \\ \text{s.t.} & \min_{x_2} & f(x_1, x_2) = (x_2 - 3)^2 \\ & \text{s.t.} & x_2^2 - x_1 \leq 0. \end{array}$$

Starting point: $(x_1, x_2) = (1, 1)$.

- `DeSilva78`: a problem from De Silva's PhD thesis [63].

$$\begin{array}{lll} \min_{x_1} & F(x_1, x_2) & = x_{11}^2 - 2x_{11} + x_{12}^2 - 2x_{12} + x_{21}^2 + x_{22}^2 \\ \text{s.t.} & \min_{x_2} & f(x_1, x_2) = (x_{21} - x11)^2 + (x_{22} - x12)^2 \\ & \text{s.t.} & 0.5 \leq x_{21} \leq 1.5 \\ & & 0.5 \leq x_{22} \leq 1.5 \end{array}$$

No starting point is given. Ours are $(x_{11}, x_{12}, x_{21}, x_{22}) = (0, 0, 1, 1)$ and $(1, 1, 1, 1)$.

- `FalkLiu95`: second example of Falk and Liu [31]. Same problem as `DeSilva78`, except the upper-level objective which is given by

$$F(x_1, x_2) = x_{11}^2 - 3x_{11} + x_{12}^2 - 3x_{12} + x_{21}^2 + x_{22}^2.$$

We use the same starting points as those of `DeSilva78` above.

- `ShimAiyo81P1`: a problem from Shimizu and Aiyoshi [61].

$$\begin{array}{lll} \min_{x_1} & F(x_1, x_2) & = x_1^2 + (x_2 - 10)^2 \\ \text{s.t.} & x_1 \leq 15, & \\ & -x_1 + x_2 & \leq 0, \\ & -x_1 \leq 0, & \\ & \min_{x_2} & f(x_1, x_2) = (x_1 + 2x_2 - 30)^2 \\ & \text{s.t.} & x_1 + x_2 \leq 20, \\ & & x_2 \leq 20, \\ & & -x_2 \leq 0. \end{array}$$

Starting point: $(x_1, x_2) = (5, 5)$.

More general nonlinear problems

- BIPA1: new problem

$$\begin{array}{lll} \min_{x_1} & F(x_1,x_2) & = (10-x_1)^3 + (10-x_2)^3 \\ \text{s.t.} & x_1 \le 15, & \\ & -x_1 + x_2 & \le 0, \\ & -x_1 \le 0, & \\ & \min_{x_2} & f(x_1,x_2) = (x_1 + 2x_2 - 15)^4 \\ & \text{s.t.} & x_1 + x_2 \le 20, \\ & & x_2 \le 20, \\ & & -x_2 \le 0. \end{array}$$

Starting point (high point): $(x_1, x_2) = (10, 10)$.

- BIPA2: new problem

$$\begin{array}{lll} \min_{x_1} & F(x_1,x_2) & = (x_1-5)^2 + (2x_2+1)^2 \\ \text{s.t.} & -x_1 \le 0, & \\ & \min_{x_2} & f(x_1,x_2) = (x_2-1)^2 - 1.5x_1x_2 + x_1^3 \\ & \text{s.t.} & -3x_1 + x_2 + 3 \le 0, \\ & & x_1 - 0.5x_2 - 4 \le 0, \\ & & x_1 + x_2 - 7 \le 0, \\ & & -x_2 \le 0. \end{array}$$

Starting point (high point): $(x_1, x_2) = (4, 0)$.

- BIPA3: new problem

$$\begin{array}{lll} \min_{x_1} & F(x_1,x_2) & = (x_1-5)^4 + (2x_2+1)^4 \\ \text{s.t.} & x_1 + x_2 - 4 & \le 0, \\ & -x_1 & \le 0, \\ & \min_{x_2} & f(x_1,x_2) = e^{-x_1+x_2} + x_1^2 + 2x_1x_2 + x_2^2 + 2x_1 + 6x_2 \\ & \text{s.t.} & -x_1 + x_2 - 2 \le 0, \\ & & -x_2 \le 0. \end{array}$$

Starting point (high point): $(x_1, x_2) = (4, 0)$.

- BIPA4: new problem

$$\begin{array}{lll} \min_{x_1} & F(x_1,x_2) = & = x_1^2 + (x_2-10)^2 \\ \text{s.t.} & x_1 + 2x_2 - 6 & \le 0, \\ & -x_1 \le 0, & \\ & \min_{x_2} & f(x_1,x_2) = x_1^3 + 2x_2^3 + x_1 - 2x_2 - x_1^2 \\ & \text{s.t.} & -x_1 + 2x_2 - 3 \le 0, \\ & & -x_2 \le 0. \end{array}$$

Starting point (high point): $(x_1, x_2) = (1.5, 2.25)$.

- BIPA5: new problem

$$
\begin{array}{lll}
\min_{x_1} & F(x_1,x_2) & = (x_{11}-x_{22})^4+(x_{21}-1)^2+(x_{21}-x_{22})^2 \\
\text{s.t.} & -x_{11}\leq 0 & \\
 & \min_{x_2} & f(x_1,x_2)=2x_{11}+e^{x_{21}}+x_{21}^2+4x_{21}+2x_{22}^2-6x_{22} \\
 & \text{s.t.} & 6x_{11}+x_{21}^2+e^{x_{22}}-15\leq 0, \\
 & & 5x_{11}+x_{21}^4-x_{22}-25\leq 0, \\
 & & -x_{21}\leq 0, \\
 & & x_{21}-4\leq 0, \\
 & & -x_{22}\leq 0, \\
 & & x_{22}-2\leq 0.
\end{array}
$$

Starting point (high point): $(x_{11},x_{21},x_{22})=(1,1,1)$.

Network design problem

The network design problem (see *e.g.* Marcotte [52] or Section 2.2 in Migdalas [53]) consists in modifying a transportation infrastructure with the objective of maximizing social welfare or minimizing design and other system costs.

In this paper, we restrict our attention to the fixed-demand network design problem and more precisely we consider the situation where the capacity of the existing links is modified. Assuming that the network has only one origin-destination pair, this problem may be formulated as follows:

$$
\begin{array}{lll}
\min\limits_{z,x} & \sum_{a\in\mathcal{A}} & [x_a t_a(x_a,z_a)+g_a(z_a)] \qquad (25a) \\
\text{s.t.} & z_a>-1 & \forall a\in\mathcal{A} \qquad (25b) \\
 & \min_x & \displaystyle\sum_{a\in\mathcal{A}}\int_0^{x_a} t_a(u,z_a)\,du \qquad (25c) \\
 & \text{s.t.} & x_a\geq 0\ \forall a\in\mathcal{A} \qquad (25d) \\
 & & \displaystyle\sum_{p\in\mathcal{P}} f_p=d \qquad (25e) \\
 & & \displaystyle x_a=\sum_{p\in\mathcal{P}}\delta_{a,p}f_p\ \forall a\in\mathcal{A} \qquad (25f)
\end{array}
$$

where we use the notation of Table 4 and

$$
\delta_{a,p}=\begin{cases} 1 & \text{if link } a \text{ belongs to route } p, \\ 0 & \text{otherwise.} \end{cases}
$$

$\mathcal{A}$	:	set of links;
z_a	:	capacity enhancement of link a ($a\in\mathcal{A}$);
x_a	:	traffic flow on link a ($a\in\mathcal{A}$);
t_a	:	travel time function for link a ($a\in\mathcal{A}$);
g_a	:	capacity function for link a ($a\in\mathcal{A}$);
$\mathcal{P}$	:	set of routes;
f_p	:	traffic flow on route p ($p\in\mathcal{P}$);
d	:	traffic demand from origin to destination.

Table 4: Notation for the network design problem.

We focused our study on the network of Figure 1, which is well-known in the traffic modeling literature and was originally presented by Braess [16]. The total traffic flow demand is assumed to be $d=6$. We also

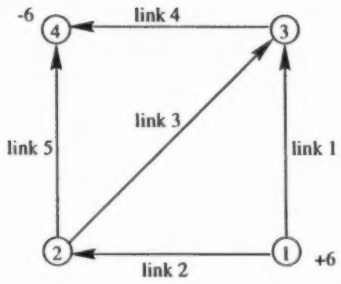

Figure 1: The Braess network considered for problems NDP1 and NDP2.

suppose that the link travel time functions are given as follows:

$$t_1(x_1, z_1) = 50 + \frac{x_1}{1+z_1}, \;\; t_2(x_2, z_2) = 10\frac{x_2}{1+z_2}, \;\; t_3(x_3, z_3) = 10 + \frac{x_3}{1+z_3},$$

$$t_4(x_4, z_4) = 10\frac{x_4}{1+z_4}, \;\; t_5(x_5, z_5) = 50 + \frac{x_5}{1+z_5},$$

while the capacity functions are assumed to take the form

$$g_a(z_a) = l_a z_a \; (a \in \mathcal{A})$$

for fixed parameters l_a ($a \in \mathcal{A}$). There are three possible routes in the network of Figure 1, namely

route 1: link 1 → link 4,
route 2: link 2 → link 5,
route 3: link 2 → link 3 → link 4,

and we have the following relationships between route and link flows:

$$\begin{aligned} x_1 &= f_1, \\ x_2 &= f_2 + f_3, \\ x_3 &= f_3, \\ x_4 &= f_1 + f_3, \\ x_5 &= f_2. \end{aligned}$$

This allows to rewrite constraints (25e) and (25f) in terms of the x_i's only:

$$\begin{aligned} x_1 + x_3 + x_5 &= 6, \\ x_2 - x_5 - x_3 &= 0, \\ x_4 - x_1 - x_3 &= 0. \end{aligned}$$

We tested BIPA on the following two problems:

- **Problem** NDP1 ($n_1 = 5, n_2 = 5, m_1 = 5, m_2 = 8$):
  we choose $l_i = 100$ ($i = 0, \ldots, 5$) and the initial values corresponding to the *high point* are

  $$z_1 = -0.7, z_2 = 0.0, z_3 = -0.999, z_4 = 0.0, z_5 = -0.7,$$

  $$x_1 = 3, x_2 = 3, x_3 = 0, x_4 = 3, x_5 = 3.$$

- **Problem** NDP2: same as NDP1 except that $l_i = 1$ ($i = 0, \ldots, 5$). The *high point* provides the following starting values:

  $$z_1 = -0.999, z_2 = 18.0, z_3 = 5.0, z_4 = 18.0, z_5 = -0.999,$$

  $$x_1 = 0, x_2 = 6, x_3 = 6, x_4 = 6, x_5 = 0.$$

Toll-setting problems

Toll-setting problems are another class of models that are frequently formulated as bilevel programs. They represent a situation where an authority or the owners of a highway system are allowed to set tolls on a subset of the links of the network while the network users wish to minimize their travel costs. An optimal toll setting is such that toll levels are not too high – otherwise the users may be deterred from using the infrastructure – though still generating profits.

The upper-level variables are the tolls, which we denote by T_a (where a belongs to the subset of tolled links $\mathcal{A}_1 \subset \mathcal{A}$), and the lower-level variables are the link traffic flows, which we still denote by x_a ($a \in \mathcal{A}$) as in the previous example.

The formulation of the toll-setting problem may be written as follows (see problem (TOP) in Labbé, Marcotte and Savard [49]):

$$\max_{T,x} \quad \sum_{a\in\mathcal{A}_1} \quad T_a x_a \tag{26a}$$

$$\text{s.t.} \quad T_a \geq l_a \quad \forall a \in \mathcal{A}_1, \tag{26b}$$

$$\min_x \quad \sum_{a\in\mathcal{A}_1} (c_a + T_a)x_a + \sum_{a\in\mathcal{A}_2} c_a x_a \tag{26c}$$

$$\text{s.t.} \quad \sum_{a\in i^+} x_a^{kl} - \sum_{a\in i^-} x_a^{kl} = 1 \text{ if } i = k,\ -1 \text{ if } i = l, \tag{26d}$$

$$\text{and 0 otherwise}, \ \forall i \in \mathcal{N}, \forall (k,l) \in \mathcal{O}\times\mathcal{D},$$

$$x_a = \sum_{(k,l)\in\mathcal{O}\times\mathcal{D}} d^{kl} x_a^{kl} \ \forall a \in \mathcal{A}, \tag{26e}$$

$$x_a^{kl} \geq 0 \ \forall a \in \mathcal{A}, \ \forall (k,l) \in \mathcal{O}\times\mathcal{D}, \tag{26f}$$

where, in addition to the notation introduced in Table 4 above, we use the symbols described in Table 5. The

c_a	:	(fixed) travel cost for link a ($a \in \mathcal{A}$), exclusive of toli;
$\mathcal{A}_2$	:	$\mathcal{A}\backslash\mathcal{A}_1$;
$\mathcal{N}$	:	set of nodes;
i^+	:	set of links exiting from node $i \in \mathcal{N}$;
i^-	:	set of links ending at node $i \in \mathcal{N}$;
$\mathcal{O}$	:	set of origin nodes;
$\mathcal{D}$	:	set of destination nodes;
x_a^{kl}	:	traffic flow from origin k to destination l on link a ($(k,l) \in \mathcal{O}\times\mathcal{D}, a \in \mathcal{A}$);
d^{kl}	:	proportion of traffic flow demand between origin k and destination l ($(k,l) \in \mathcal{O}\times\mathcal{D}$);
l_a	:	lower bound on toll for link a ($a \in \mathcal{A}_1$).

Table 5: Additional notation for the toll-setting problem.

two problem instances we solved with BIPA are built as follows (note that the tolled links are represented with dashed lines on the accompanying figures):

- **Problem** TOLL1 ($n_1 = 3, n_2 = 8, m_1 = 3, m_2 = 13$):
  We consider the network and the costs c_a ($a \in \mathcal{A}$) represented on Figure 2. Nodes 1 and 5 constitute its unique origin-destination pair. Other data values include the following items
  - $d^{(1,5)} = 1$;
  - $\mathcal{A}_1 = \{3, 4, 8\}$,
  - $\mathcal{A}_2 = \{1, 2, 5, 6, 7\}$,

- upper-level variables: T_3, T_4 and T_8,
- lower-level variables: x_i, $i = 1, \ldots, 8$,
- starting point: $(T_3, T_4, T_8, x_1, \ldots, x_8) = (0, 5, 5, 0, 1, 0, 0, 0, 1, 0, 0)$.

This problem was originally presented in Labbé *et al* [49].

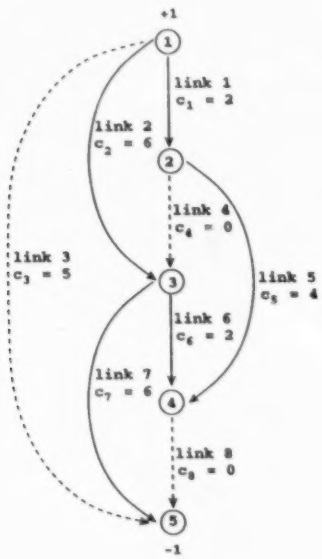

Figure 2: Network for problem TOLL1.

- **Problem** TOLL2 ($n_1 = 3, n_2 = 18, m_1 = 3, m_2 = 28$):
  The reference network is depicted on Figure 3. There are two origin-destination pairs, namely (O_1, D_1) and (O_2, D_2), with further inputs given as follows:
  - $d^{(O_1,D_1)} = d^{(O_2,D_2)} = 1$;
  - $\mathcal{A}_1 = \{1, 2, 3\}$,
  - $\mathcal{A}_2 = \{4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$,
  - upper-level variables: T_1, T_2 and T_3,
  - lower-level variables: x_i^1, x_i^2 $(i = 1, 2, 3)$ and x_j $(j = 4, \ldots, 15)$, where x_i^k denotes the proportion of flow on link i related to the demand of the k-th origin-destination pair,
  - starting point: $(T_1, T_2, T_3) = (0, 0, 0)$ and $(x_1^1, x_1^2, x_2^1, x_2^2, x_3^1, x_3^2, x_4, \ldots, x_{15}) = (0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1)$.

- **Problem** TOLL3: same as TOLL2 except that the demand for O-D pair (O_2, D_2) is 10 instead of 1. The starting point is $(T_1, T_2, T_3) = (0, 0, 0)$ and $(x_1^1, x_1^2, x_2^1, x_2^2, x_3^1, x_3^2, x_4, \ldots, x_{15}) = (0, 0, 0, 10, 1, 0, 1, 0, 0, 0, 10, 0, 0, 0, 1, 0, 0, 10)$.

- **Problem** TOLL4 ($n_1 = 2, n_2 = 4, m_1 = 2, m_2 = 6$):
  This corresponds to an example (Figure 4.3, page 123) in Brotcorne [17], whose network is depicted on Figure 4 and contains two O-D pairs, $(1, 2)$ and $(3, 4)$.

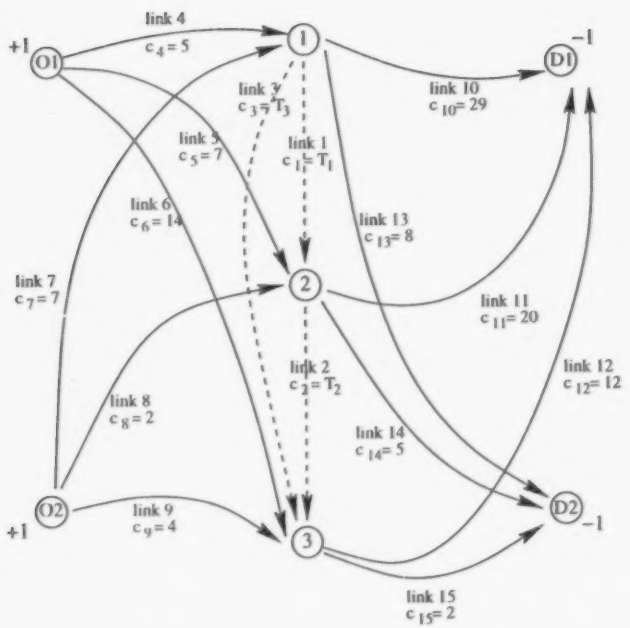

Figure 3: Network for problem TOLL2.

- $d^{(1,2)} = d^{(5,6)} = 1$;
- $\mathcal{A}_1 = \{4, 6\}$,
- $\mathcal{A}_2 = \{1, 2, 3, 5, 7\}$,
- upper-level variables: T_4 and T_6,
- lower-level variables: $x_1, x_7, x_2 = x_4^1 = x_3$ and $x_5 = x_4^2 = x_6$,
- starting point: $(T_4, T_6, x_1, x_2, x_5, x_7) = (0, 0, 0, 0, 0, 0)$.

Note that this problem does not involve lower-bound constraints of the type (26b), therefore allowing for negative tolls.

- **Problem** TOLL5 ($n_1 = 1, n_2 = 4, m_1 = 1, m_2 = 6$):
  This is another example from Brotcorne [17] (Figure 4.4, page 125), corresponding to the network of Figure 5, with O-D pairs $(1, 2)$ and $(3, 4)$.
  - $d^{(1,2)} = d^{(5,6)} = 1$;
  - $\mathcal{A}_1 = \{4\}$,
  - $\mathcal{A}_2 = \{1, 2, 3, 5, 6, 7\}$,
  - upper-level variable: T_4,
  - lower-level variables: $x_1, x_7, x_2 = x_4^1 = x_3$ and $x_5 = x_4^2 = x_6$,
  - starting point: $(T_4, x_1, x_2, x_5, x_7) = (0, 0, 0, 0, 0)$.

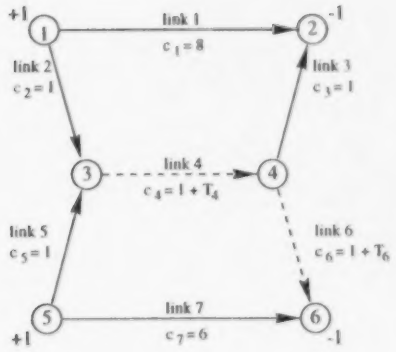

Figure 4: Network for problem TOLL4.

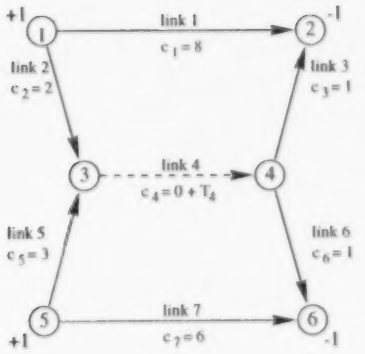

Figure 5: Network for problem TOLL5.

MPEC

The following problem (`Outrata94P1`) was introduced by Outrata [54].

$$\begin{array}{lll} \min_{x_1} & F(x_1,x_2) & \\ \text{s.t.} & 0 \le x_1 \le 10 & \\ & \min_{x_2} & f(x_1,x_2) \\ & \text{s.t.} & -0.333x_{21} + x_{22} - 1 + 0.1x_1 \le 0, \\ & & x_{21}^2 + x_{22}^2 - 9 - 0.1x_1 \le 0, \\ & & -x_{21} \le 0, \\ & & -x_{22} \le 0, \end{array}$$

where

$$f(x_1,x_2) = \frac{1}{2}\left((1+0.2x_1)x_{21}^2 + (1+0.1x_1)x_{22}^2\right) - (3+1.333x_1)x_{21} - x_1x_{22},$$

and

$$F(x_1,x_2) = \frac{1}{2}(x_{21}-3)^2 + \frac{1}{2}(x_{22}-4.0)^2$$

Two initial values for x_1 are proposed in [54]: $x_1 = 0$ and $x_1 = 10$.

References

[1] E. Aiyoshi and K. Shimizu. Hierarchical decentralized systems and its new solution by a barrier method. *IEEE Transactions on Systems, Man, and Cybernetics*, 11:444–449, 1981.

[2] E. Aiyoshi and K. Shimizu. A solution method for the static constrained Stackelberg problem via penalty method. *IEEE Transactions on Automatic Control*, 29:1111–1114, 1984.

[3] F. A. Al-Khayal, R. Horst, and P. M. Pardalos. Global optimization of concave functions subject to quadratic constraints : an application in nonlinear bilevel programming. *Annals of Operations Research*, 34:125–147, 1992.

[4] G. Anandalingam and T. Friesz. Hierarchical optimization: an introduction. *Annals of Operations Research*, 34:1–11, 1992.

[5] A. Auslender. *Optimisation: Méthodes numériques*. Masson, Paris, 1976.

[6] J. F. Bard. Convex two-level optimization. *Mathematical Programming*, 40:15–27, 1988.

[7] J. F. Bard. *Practical bilevel optimization – Algorithms and applications*. Kluwer Academic Publishers, 1998.

[8] J. F. Bard and J. Falk. An explicit solution to the multi-level programming problem. *Computers and Operations Research*, 9:77–100, 1982.

[9] J. F. Bard and J. T. Moore. A branch and bound algorithm for the bilevel programming problem. *SIAM Journal of Scientific and Statistical Computing*, 11(2):281–292, 1990.

[10] O. Ben-Ayed. Bilevel linear programming. *Computers and Operations Research*, 20:485–501, 1993.

[11] Z. Bi, P. H. Calamai, and A. R. Conn. An exact penalty function approach for the linear bilevel programming problem. Technical Report #167-O-310789, Department of Systems Design Engineering, University of Waterloo, 1989.

[12] Z. Bi, P. H. Calamai, and A. R. Conn. An exact penalty function approach for the nonlinear bilevel programming problem. Technical Report #180-O-170591, Department of Systems Design Engineering, University of Waterloo, 1991.

[13] W. Bialas and M. Karwan. Two-level linear programming. *Management Science*, 30:1004–1020, 1984.

[14] W. Bialas, M. Karwan, and J. Shaw. A parametric complementarity pivot approach for two-level linear programming. Technical Report 80-2, State University of New York at Buffalo, Operations Research Program, 1980.

[15] J. Bracken and J. McGill. Mathematical programs with optimization problems in the constraints. *Operations Research*, 21:37–44, 1973.

[16] D. Braess. Über ein Paradox der Verkehrsplannung. *Unternehmenstorchung*, 12:258–268, 1968.

[17] L. Brotcorne. *Approches opérationnelles et stratégiques des problèmes de trafic routier*. PhD thesis, Université Libre de Bruxelles, Brussels, Belgium, 1998.

[18] P. H. Calamai and L. N. Vicente. Generating linear and linear-quadratic bilevel programming problems. *SIAM Journal on Scientific Computing*, 14:770–782, 1993.

[19] P. H. Calamai and L. N. Vicente. Algorithm 728: FORTRAN subroutines for generating quadratic bilevel programming problems. *ACM Transactions on Mathematical Software*, 20:120–123, 1994.

[20] P. H. Calamai and L. N. Vicente. Generating quadratic bilevel programming test problems. *ACM Transactions on Mathematical Software*, 20:103–119, 1994.

[21] P. H. Calamai, L. N. Vicente, and J. J. Júdice. A new technique for generating quadratic programming test problems. *Mathematical Programming*, 61:215–231, 1993.

[22] W. Candler and R. Norton. Multilevel programing. Technical Report 20, World Bank Development Research Center, Washington D.C., 1977.

[23] W. Candler and R. Townsley. A linear two-level programming problem. *Computers and Operations Research*, 9:59–76, 1982.

[24] L. M. Case. *An l_1 penalty function approach to the nonlinear bilevel programming problem*. PhD thesis, University of Waterloo, Ontario, Canada, 1999.

[25] B. Colson. Mathematical programs with equilibrium constraints and nonlinear bilevel programming problems. Master's thesis, Department of Mathematics, FUNDP, Namur, Belgium, September 1999.

[26] B. Colson. BIPA (BIlevel Programming with Approximation methods): software guide and test problems. Technical report, Centre de Recherche sur les Transports, Université de Montréal, Montréal, QC, Canada, 2002. Forthcoming.

[27] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. *Trust-region methods*. SIAM, 2000.

[28] S. Dempe. A necessary and a sufficient optimality condition for bilevel programming problems. *Optimization*, 25:341–354, 1992.

[29] T. Edmunds and J. F. Bard. Algorithms for nonlinear bilevel mathematical programs. *IEEE transactions on Systems, Man, and Cybernetics*, 21(1):83–89, 1991.

[30] F. Facchinei, H. Jiang, and L. Qi. A smoothing method for mathematical programs with equilibrium constraints. Technical Report R03-96, Università di Roma "La Sapienza", Dipartimento di Informatica e Sistemistica, 1996.

[31] J. E. Falk and J. Liu. On bilevel programming, Part I : general nonlinear cases. *Mathematical Programming*, 70(1):47–72, 1995.

[32] R. Fletcher and S. Leyffer. Nonlinear programming without a penalty function. Numerical Analysis Report NA 97-2, Department of Mathematics, University of Dundee, Dundee, DD1 4HN. UK, September 1997. To appear in Mathematical Programming.

[33] R. Fletcher and S. Leyffer. Numerical experience with solving MPECs by nonlinear programming methods. Numerical Analysis Report, Department of Mathematics, University of Dundee, Dundee, DD1 4HN. UK, 2002. In preparation.

[34] R. Fletcher, S. Leyffer, D. Ralph, and S. Scholtes. Local convergence of SQP methods for Mathematical Programs with Equilibrium Constraints. Numerical Analysis Report NA-209, Department of Mathematics, University of Dundee, Dundee, DD1 4HN. UK, April 2002.

[35] C. A. Floudas, P. M. Pardalos, C. S. Adjiman, W. R. Esposito, Z. H. Gumus, S. T. Harding, J. L. Klepeis, C. A. Meyer, and C. A. Schweiger. *Handbook of test problems in local and global optimization*. Kluwer, 1999.

[36] J. Fortuny-Amat and B. McCarl. A representation and economic interpretation of a two-level programming problem. *Journal of the Operational Research Society*, 32:783–792, 1981.

[37] M. Fukushima. Equivalent differentiable optimization problems and descent methods for asymmetric variational inequality problems. *Mathematical Programming*, 53:99–110, 1992.

[38] M. Fukushima and J.-S. Pang. Complementarity constraint qualifications and simplified B-stationarity conditions for mathematical programs with equilibrium constraints. *Computational Optimization and Applications*, 13:111–136, 1999.

[39] P. Hansen, B. Jaumard, and G. Savard. New branch-and-bound rules for linear bilevel programming. *SIAM Journal on Scientific and Statistical Computing*, 13(5):1194–1217, September 1992.

[40] S. Hsu and U. Wen. A review of linear bilevel programming problems. In *Proceedings of the National Science Council, Republic of China, Part A: Physical Science and Engineering*, volume 13, pages 53–61, 1989.

[41] ILOG CPLEX Division. *CPLEX User's Guide*.

[42] Y. Ishizuka and E. Aiyoshi. Double penalty method for bilevel optimization problems. *Annals of Operations Research*, 34:73–88, 1992.

[43] B. Jaumard, G. Savard, and J. Xiong. An exact algorithm for convex quadratic bilevel programming. Draft paper, Ecole Polytechnique de Montréal, January 2000.

[44] H. Jian and D. Ralph. Smooth sqp methods for mathematical programs with nonlinear complementarity constraints. Manuscript, Department of Mathematics and Statistics, University of Melbourne, December 1997.

[45] J. Júdice and A. Faustino. The solution of the linear bilevel programming problem by using the linear complementarity problem. *Investigação Operacional*, 8:77–95, 1988.

[46] J. Júdice and A. Faustino. A sequential LCP method for bilevel linear programming. *Annals of Operations Research*, 34:89–106, 1992.

[47] C. D. Kolstad. A review of the literature on bi-level mathematical programming. Technical Report LA-10284-MS, Los Alamos National Laboratory, Los Alamos, New Mexico, 1985.

[48] C. D. Kolstad and L. S. Lasdon. Derivative estimation and computational experience with large bilevel mathematical programs. *Journal of Optimization Theory & Applications*, 65:485–499, 1990.

[49] M. Labbé, P. Marcotte, and G. Savard. A bilevel model of taxation and its applications to optimal highway pricing. *Management Science*, 44:1595–1607, 1998.

[50] S. Leyffer. MacMPEC – Ampl collection of Mathematical Programs with Equilibrium Constraints. `http://www.maths.dundee.ac.uk/ sleyffer/MacMPEC/index.html`.

[51] Z.-Q. Luo, J.-S. Pang, and D. Ralph. *Mathematical programs with equilibrium constraints*. Cambridge University Press, 1996.

[52] P. Marcotte. Network design problem with congestion effects : A case of bilevel programming. *Mathematical Programming*, 34:142–162, 1986.

[53] A. Migdalas. Bilevel programming in traffic planning: models, methods and challenge. *Journal of Global Optimization*, 7:381–405, 1995.

[54] J. Outrata. On optimization problems with variational inequality constraints. *SIAM Journal on Optimization*, 4(2):340–357, 1994.

[55] J. V. Outrata, M. Kocvara, and J. Zowe. *Nonsmooth approach to optimization problems with equilibrium constraints*. Kluwer Academic Publishers, 1998.

[56] D. Ralph. Optimization with equilibrium constraints: A piecewise SQP approach. Manuscript, Department of Mathematics and Statistics, University of Melbourne, November 1998.

[57] G. Savard. *Contribution à la programmation mathématique à deux niveaux*. PhD thesis, Ecole Polytechnique de Montréal, Université de Montréal, April 1989.

[58] G. Savard and J. Gauvin. The steepest descent direction for the nonlinear bilevel programming problem. *Operations Research Letters*, 15:265–272, 1994.

[59] S. Scholtes. Convergence properties of a regularisation scheme for mathematical programs with complementarity constraints. *SIAM Journal on Optimization*, 11:918–936, 2001.

[60] S. Scholtes and M. Stöhr. Exact penalization of mathematical programs with equilibrium constraints. *SIAM Journal on Control and Optimization*, 37(2):617–652, 1999.

[61] K. Shimizu and E. Aiyoshi. A new computational method for Stackelberg and min-max problems by use of a penalty method. *IEEE Transactions on Automatic Control*, 26:460–466, 1981.

[62] K. Shimizu, Y. Ishizuka, and J. F. Bard. *Nondifferentiable and two-level mathematical programming*. Kluwer Academic Publishers, 1997.

[63] A. H. De Silva. *Sensitivity Formulas for Nonlinear Factorable Programming and their Application to the Solution of an Implicitly Defined Optimization Model of US Crude Oil Production*. PhD thesis, George Washington University, 1978.

[64] P. Spellucci. *DONLP2 users guide*. Technical University at Darmstadt, Department of Mathematics, 64829 Darmstadt, Germany.

[65] P. Spellucci. A new technique for inconsistent problems in the SQP method. *Math. Meth. of Oper. Res.*, 47:355–400, 1998.

[66] P. Spellucci. An SQP method for general nonlinear programs using only equality constrained subproblems. *Mathematical Programming*, 82:413–448, 1998.

[67] H. Tuy, A. Migdalas, and P. Värbrand. A global optimization approach for the linear two-level program. *Journal of Global Optimization*, 3:1–23, 1993.

[68] L. N. Vicente and P. H. Calamai. Bilevel and multilevel programming: a bibliography review. *Journal of Global Optimization*, 5(3):291–306, 1994.

[69] L. N. Vicente, P. H. Calamai, and J. J. Júdice. Generation of disjointly constrained bilinear programming test problems. *Computational Optimization and applications*, 1:299–306, 1992.

[70] L. N. Vicente, G. Savard, and J. J. Júdice. Discrete linear bilevel programming problem. *Journal of Optimization Theory and Applications*, 89:597–614, 1996.

[71] U. Wen and S. Hsu. Linear bi-level programming problems – a review. *Journal of the Operational Research Society*, 42:125–133, 1991.